

DepthSynth: Real-Time Realistic Synthetic Data Generation from CAD Models for 2.5D Recognition

Benjamin Planche
Siemens AG, Germany

benjamin.planche@siemens.com

Ziyan Wu
Siemens Corporation, USA

ziyan.wu@siemens.com

Kai Ma, Shanhui Sun, Stefan Kluckner, Terrence Chen
Siemens Healthineers, USA

{kai.ma, shanhui.sun, stefan.kluckner, terrence.chen}@siemens.com

Andreas Hutter, Sergey Zakharov
Siemens AG, Germany

{ andreas.hutter, sergey.zakharov } @siemens.com

Harald Kosch
University of Passau, Germany

harald.kosch@uni-passau.de

Jan Ernst
Siemens Corporation, USA

jan.ernst@siemens.com

Abstract

Recent progress in computer vision has been dominated by deep neural networks trained with large amount of labeled data. Collecting and annotating such datasets is however a tedious, and in some contexts impossible task; hence a recent surge in approaches that rely solely on synthetically generated data from 3D models for their training. For depth images however, the discrepancies with real scans noticeably affect the performance of such methods. In this paper, we propose an innovative end-to-end framework which simulate the whole mechanism of these devices, synthetically generating realistic depth data from 3D CAD models by comprehensively modeling vital factors such as sensor noise, material reflectance, surface geometry, etc. Besides covering a wider range of sensors than state-of-the-art methods, the proposed one also results in more realistic data. Going further than previous works, we not only qualitatively evaluate the generated scans, but also quantitatively measure through extensive experiments and comparisons how they impact the training of neural network algorithms for different 3D recognition tasks, demonstrating how our pipeline seamlessly integrates such architectures; and how it consistently and significantly enhances their performance—irrespective of the selected feature space or intermediate representations.

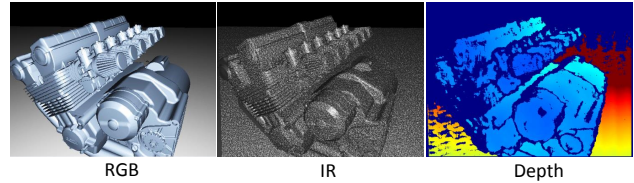


Figure 1. Synthetic sample generated by the proposed pipeline.

1. Introduction

Understanding the three dimensional shape or spatial layout of a real-world object captured in a 2D image has been a classic computer vision problem for decades [5, 12, 45]. However, with the advent of low-cost depth cameras—specifically structured light 3D scanners [17] e.g. Microsoft Kinect, Occipital Structure, Intel RealSense—its focus has seen a substantial paradigm shift. What in the past revolved around interpretation of raw pixels in 2D projections has now become the analysis of real-valued depth (2.5D) data. This has drastically increased the scope of practical applications ranging from recovering 3D geometry of complex surfaces [24, 29] to real-time recognition of human actions [34], inspiring research in automatic detection [9, 32, 35, 38], classification [19, 26, 32, 36–38] and pose estimation [9, 23, 33] of real-world objects.

While real acquired data samples were commonly used for comparison and training, a large number of these recent studies decompose the problems to matching acquired

depth images of real-world objects to synthetic ones exhaustively rendered from a database of pre-existing CAD models [9, 23, 32, 36–38]. Since there is no theoretical upper bound on obtaining synthetically generated images to either train complex models for pose classification [26, 33, 40, 44] or fill large databases for retrieval tasks [11, 42], research continues to gain impetus in this direction.

Despite the obvious simplicity of the above flavor of approaches, their performance is often restrained by the lack of *realism* (discrepancy with real data) or *variability* (limited configurability) of their synthetic depth image generation. As a workaround, some approaches fine-tune their systems on a small set of real scans to improve their performance [43]. In many cases however, access to real data is too scarce to bridge the discrepancy gap. Other generic classification methods try instead to post-process the real images to clear some of their noise, making them more similar to synthetic data, but losing details in the process [?, 44]. These details can be crucial for many other tasks such pose estimation or fine-grained object recognition. A reasonably practical approach to address this problem globally is thus to generate larger amount of synthetic data, and in such a way that they mimic captured depth images. This is however a non-trivial problem, as it is extremely difficult to exhaustively enumerate all possible physical variations of a given object—including surface geometry, material reflectance, deformation, etc. In this paper, we attempt to address these challenges in order to synthetically generate realistic depth data, replicating realistic capture scenarios (Figure 1), thereby facilitating robust 2.5D applications. The key contributions are as follows: (a) we introduce *DepthSynth*, an end-to-end pipeline to synthetically generate depth images from 3D models by virtually and comprehensively reproducing the sensors mechanisms, making it insensitive to the ulterior choice of algorithm or feature space; (b) we systematically evaluate and compare the quality of the resulting images with theoretical models and other modern simulation methods; (c) we demonstrate the effectiveness and flexibility of our tool by pairing it to a state-of-the-art method for two 2.5D recognition tasks.

The rest of the paper is organized as follows. In Section 2, we provide a survey of pertinent work to the interested reader. Next, in Section 3, we introduce our synthetic data generation framework, detailing each of its steps. In Section 4, we elaborate on our experimental protocol; first comparing the sensing errors induced by our tool to those of experimental depth images and theoretical models; then briefing on the pose estimation and classification experiments used as examples, providing comprehensive results to demonstrate the usefulness of our method. We finally conclude with insightful discussions in Section 5.

2. Related Work

With the popular advocacy of 2.5D/3D sensor for vision applications, depth data is the support of active research within computer vision. We emphasize on recent approaches which employ synthetic data, and present previous methods to generate such 2.5D images from CAD models.

Depth-based Methods and Synthetic Data Crafting features to efficiently detect objects, discriminate them, evaluate their poses, etc. has long been a tedious task for computer vision researchers. With the rise of machine learning algorithms, these existing models have been complemented [12, 33, 39], before being almost fully replaced by statistically-learned representations. Multiple recent approaches based on deep convolutional neural networks unequivocally outshone previous methods [26, 40, 41, 44, 44], taking advantage of growing image datasets (such as *ImageNet* [7]) for their extensive training. As a matter of fact, collecting and accurately labeling large amount of real data is however an extremely tedious task, especially when 3D poses are considered for ground truth.

In order to tackle this limitation, and concomitantly with the emergence of 3D model databases, renewed efforts [31, 41] were put into the synthetic extension of image or depth scan datasets, by applying various deformations and noise to the original pictures or by rendering images from missing viewpoints. These augmented datasets were then used to train more flexible estimators. Among other deep learning-based methods for class and pose retrieval recently proposed [26, 40, 44], Wu *et al.* *3D Shapenets* [44] and Su *et al.* *Render-for-CNN* [41] methods are two great examples of a second trend: using the *ModelNet* [44] and *ShapeNet* [4] 3D model datasets they put together, they let their networks learn features from this purely synthetic data, achieving consistent results in object registration, next-best-view prediction or pose estimation.

Diving further into the problem of depth-based object classification and pose estimation chosen as illustration in this paper, Wohlhart *et al.* [43] recently developed a scalable process addressing a two-degree-of-freedom pose estimation problem. Their approach evaluates the similarity between descriptors learned by a Convolutional Neural Network (CNN) with Euclidean distance, followed by nearest neighbor search. They trained their network with real captured data, but also simplistic synthetic images rendered from 3D models. In our work, this framework is extended to recognizing 3D pose with six degrees of freedom (6-DOF), and fed only with realistic synthetic images from *DepthSynth*. This way we achieve a significantly higher flexibility and scalability of the system, as well as a more seamless application to real-world use cases.

Synthetic Depth Image Generation Early research along this direction involves the work of [3, 25], wherein search

based on 3D representations are introduced. More recently, Rozantsev *et al.* presented a thorough method for generating synthetic images [32]. Instead of focusing on making them look similar to real data for an empirical eye, they worked on a similarity metric based on the features extracted during the machine training. However, their model is tightly bound to properties impairing regular cameras (e.g. lighting and motion blur), which can't be applied to depth sensors.

Su *et al.* worked concurrently on a similar pipeline [41], optimizing a synthetic RGB image renderer for the training of CNNs. While working on finding the best compromise between quality and scalability, they notice the ability CNNs have to *cheat* at learning from too simplistic images (e.g. by using the constant lighting to deduce the models poses, or by relying too much on contours for pictures rendered without background, etc.). Their pipeline has thus been divided into three steps: the rendering from 3D models, using random lighting parameters; the alpha composition with background images sampled from the SUN397 dataset [14]; and randomized cropping. By outperforming state-of-the-art pose estimation methods with their own one trained on synthetic images, they demonstrated the benefits such pipelines can bring to computer vision.

Composed of similar steps as the method above, *Depth-Synth* can also be compared to the one by Landau *et al.* [20, 21], reproducing the Microsoft Kinect's behavior by simulating the infrared capture and stereo-matching process. Though their latter step inspired our own work, we opted for a less empirical, more exhaustive and generic model for the simulated projection and capture of pattern(s).

For the sake of completeness, tools such as *BlenSor* [13], or *pcl::simulation* [1, 8] should also be mentioned. However, such simulators were implemented to help testing vision applications, and rely on a more simplistic modeling of the sensors, e.g. ignoring reflectance effects or using fractal noise for approximations.

3. Methodology

Our end-to-end pipeline for low-latency generation of realistic depth images from 3D CAD data covers various types of 3D/2.5D sensors including single-shot/multi-shot structured light sensors, as well as Time-of-Flight (ToF) sensors are relatively simpler than structured-light ones to simulate, using a sub-set of the pipeline's components (e.g. i.i.d. per-pixel noise based on distance and object surface material, etc.). From here, we will mostly focus on single-shot sensors since they are the most popular type of devices used by the research community, e.g. Microsoft Kinect, Occipital Structure and Xtion Pro Live. This proposed pipeline can be defined as a sequence of procedures directly inspired by the underlying mechanism of the sensors we are simulating; i.e. from pattern projection and capture, followed by pre-processing and depth reconstruction using the

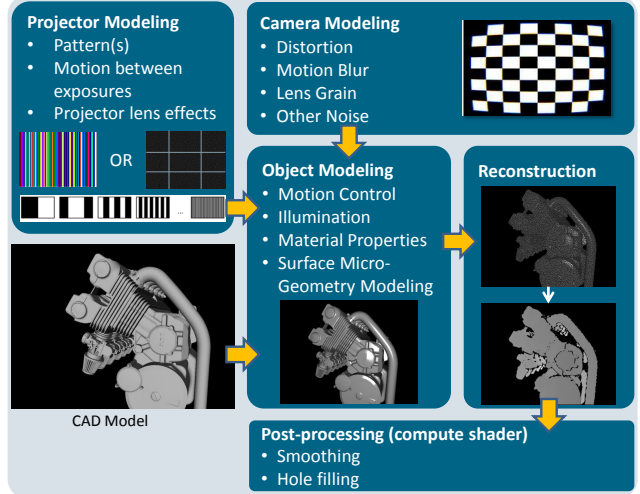


Figure 2. Representation of the end-to-end pipeline for synthetic depth scans generation.

acquired image and original pattern, to post-processing; as illustrated in Figure 2.

3.1. Understanding the Noise Causes

To realistically generate synthetic depth data, we need first to understand the causes behind the various kinds of noise one can find in the scans generated by real structured light sensors. We thus analyzed the different kinds of noise impairing structured light sensors, and their sources and characteristics. This study highlighted how each step of the sensing process introduces its own artifacts. During the initial step of projection and capture of the pattern(s), noise can be induced by the lighting and material properties of the surfaces (too low or strong reflection of the pattern), by the composition of the scene (e.g. pattern's density per unit area drops quadratically with increasing distance causing axial noise, non-uniformity at edges causing lateral noise, and objects obstructing the path of the emitter, of the camera or both causing shadow noise), or by the sensor structure itself (structural noise due to its low spatial resolution or the wrapping of the pattern by the lenses). Further errors and approximations are then introduced during the block-matching and hole-filling operations—such as structural noise caused by the disparity-to-depth transform, band noise caused by windowing effect during block correlation, or growing step size as depth increases during quantization.

By using the proper rendering parameters and applying the described depth data reconstruction procedure, the proposed synthetic data generation pipeline is able to exhaustively induce the aforementioned types of noise, unlike other state-of-the-art depth data simulation methods, as highlighted by the comparison in Table 1.

Table 1. Comparison between *BlenSor*, Landau’s pipeline and ours on different types of sensor noise.

Type of Noise	BlenSor [13]	Landau’s [20, 21]	Ours
Axial and Lateral Noise	Yes	Yes	Yes
Specular Surface	Yes	No	Yes
Non-specular Surface	No	No	Yes
Structural Noise	No	Partial	Yes
Lens Distortion and Effects	No	No	Yes
Quantization Step Noise	No	Yes	Yes
Motion and Rolling Shutter	No	No	Yes
Shadow	No	Partial	Yes

3.2. Pattern Projection and Capture

In the first part of the presented pipeline, a simulation platform is used to reproduce the realistic pattern projection and capture mechanism. Thanks to an extensive set of parameters, this platform is able to behave like a large panel of depth sensors. Indeed, any kind of pattern can first be provided as an image asset for the projection, in order to adapt to the single-shot—or multi-shot—depth sensing device one wants to simulate. Moreover, the intrinsic and extrinsic parameters of the camera and projector are configurable.

Our procedure covers both the full calibration of real structured light sensors and the reconstruction of their projected pattern with the help of an extra camera. Once the original pattern obtained, our pipeline automatically generates a square binary version of it, followed by other different ones later used as reference in the block matching procedure according to the image resolution of the camera.

Once in possession of these parameters, they can be handed to the 3D platform to initialize the simulation. The 3D model must then be provided, along with its material(s). Even though not all 3D models come with realistic textures, the quality of the synthetic results highly depends on such characteristics—especially on their reflectance definition.

Indeed, given also a list of viewpoints, the platform will perform each pattern capture and projection, simulating realistic illumination sources and shadows, taking into account surface and material characteristics. In addition to the object model, the 3D scene is thus populated with:

- A spot light projector, using the desired high resolution pattern as light cookie;
- A camera model, set up with the intrinsic and extrinsic parameters of the real sensor, separated from the projector by the provided baseline distance in the horizontal plan of the simulated device;
- Optionally additional light sources, to simulate the effect of environmental illuminations;
- Optionally other 3D models (e.g. ground, occluding objects, etc.), to ornament the scene.

By using a virtual light projector using the provided pattern(s), and a virtual camera with the proper optical characteristics, we reproduce the exact light projection / capture which the real devices are based on, obtaining out of the 3D

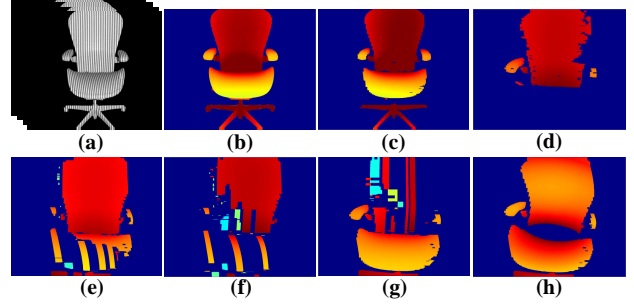


Figure 3. Examples of synthetic depth data generation for multi-shot structured light sensors. (a) Rendering of projected patterns under realistic lighting and surface materials. (b) Ideal depth data. (c) Reconstructed depth map with *DepthSynth*. (d) Depth map with strong ambient light in the environment. (e) Motion between exposures with constant speed of 5 cm/s. (f) Motion between exposures with constant speed of 10 cm/s. (g) Vibrations applied to the sensor, synchronized to its pattern-switching mechanism, and with amplitude of 2 cm. (h) Rolling shutter effect when moving with constant speed of 10 cm/s in the direction parallel to the horizontal axis of the image.

engine a captured image with the chosen resolution, similar to the intermediate output of the devices (e.g. the infrared captures from Microsoft Kinect or Occipital Structure).

3.3. Pre-processing of Pattern Captures

This intermediate result, captured in real-time by the virtual camera, is then pre-processed (fed into a *compute shader* layer), in order to get closer to the original quality, impinged by imaging sensor noise.

In this module, noise effects are added, including radial and tangential lens distortion, lens scratch and grain, motion blur and independent and identically distributed random noise.

3.4. Stereo-matching

Relying on the principles of stereo vision, the rendered picture is then matched with its reference pattern, in order to extract the depth information from their disparity map. The emitted pattern and the resulting capture from the sensor are here used as the stereo stimuli, with these two virtual eyes (the projector and the camera) being separated by the *baseline* distance b . The depth value z is then a direct function of the disparity d with $z = f \cdot b / d$, where f is the focal length in pixel of the receiver.

The disparity map is computed by applying a block-matching process using small *Sum of Absolute Differences* (SAD) windows to find the correspondences [16], sliding the window along the epipolar line. The function value of

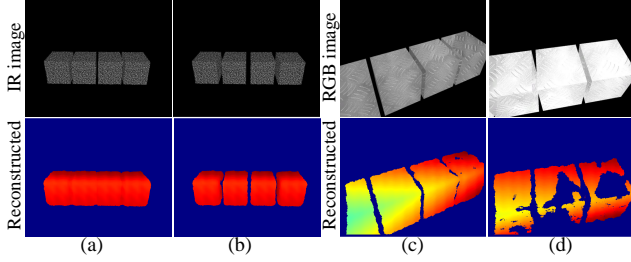


Figure 4. **Examples of synthetic image pairs for target objects (4 cubes of 0.2m^3) with different placements and surface materials.** Cubes in are separated by 5 cm in (a), and by 10 cm in (b, c, d). Cubes have fully diffused material in (a, b), 20% reflective material in (c), and 50% reflective material in (d).

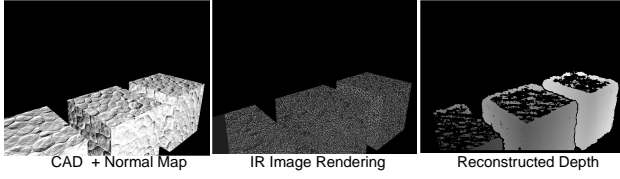


Figure 5. **Example of the effects of surface conditions on the simulation,** applying a textured normal map to the target objects.

SAD for the location (x, y) on the captured image is:

$$F_{SAD}(u, v) = \sum_j \sum_i^{w-1} |I_s(x+i, y+j) - I_t(x+u+i, y+v+j)| \quad (1)$$

where w is the window size, I_s the image from the camera, and I_t the pattern image. The matched location on the pattern image can be obtained by:

$$(u_m, v_m) = \underset{(u,v)}{\operatorname{argmin}} F_{SAD}(u, v) \quad (2)$$

The disparity value d can be computed by:

$$d = \begin{cases} u_m - x & \text{horizontal stereo} \\ v_m - y & \text{vertical stereo} \end{cases} \quad (3)$$

Based on pixel offsets, each disparity value is an integer. Refinement is done by interpolating between the closest matching block and its neighbors, achieving a sub-pixel accuracy. Given the direct relation between z and d , the possible disparity values are directly bound to the sensor’s operational depth range, limiting the search range itself.

3.5. Post-processing of Depth Scans

Finally, the depth maps undergo post-processing through another *compute shader* layer, where they will be smoothed and trimmed according to the measurement range from the sensor’s specifications. Imitating once more the operations done by the real devices, a hole-filling step can be performed to reduce the missing data proportion.

Figures 4 and 5 show how *DepthSynth* is able to realistically reproduce the spatial sensitivity of the devices or the impact of surface materials. In the same way, Figure 3(d)-

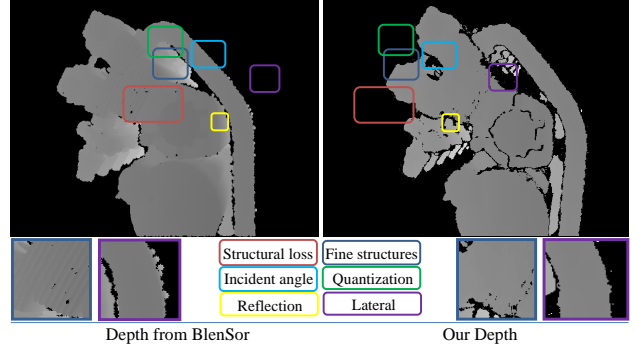


Figure 6. **Detailed comparison with *BlenSor* [13]** highlighting the salient differences, according to the study in Subsection 3.1.

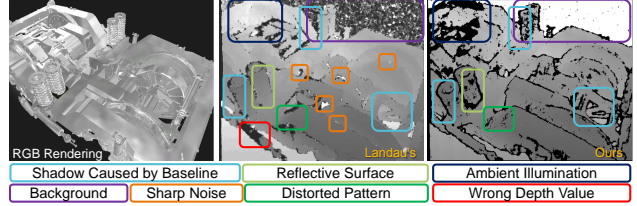


Figure 7. **Detailed comparison with Landau’s solution [20, 21],** based on the study in Subsection 3.1.

(h) reveals how the data quality of simulated multi-shot structured light sensors is highly sensitive to motion—an observation in accordance to our expectations. As highlighted in Figure 6 and Figure 7 with the visual comparisons between *DepthSynth*, *BlenSor* and Landau’s method, the latter methods aren’t sensitive to some of these realistic effects during capture, or preserve all fine details which are actually smoothed-out up to the window-size by the devices in block-matching. Our determination to closely reproduce the whole process performed by the devices paid off in terms of noise quality.

3.6. Background Blending

Background is one of the major elements one would expect in real depth images. Most of the existing synthetic depth data generation pipelines chose to ignore background addition (e.g. by alpha-compositing) which may cause significant discrepancy between synthetic data and real data, and bias the learner. Background modeling is hence another key component in our simulation pipeline. The types of backgrounds covered in our study include: (1) Static synthetic background from predefined geometry; (2) Dynamic synthetic background from predefined geometry and motion; (3) Synthetic background with large amount of random primitive shapes; (4) Real captured background (e.g. from public datasets).

Once optimized through GPU implementation, the overall process can generate realistic scans and their ground truth (noiseless scan, label image and formatted pose

matrix) about 10 scans per second on a middle-range computer (Intel E5-1620 v2, 16 GB RAM, NVidia K4200), fast enough to build large datasets.

4. Experiments and Results

To demonstrate the accuracy and practicality of our simulation tool, we first analyze in Subsection 4.1 the depth error it induces when simulating the Kinect device, comparing with other simulation tools, experimental depth images and theoretical models for this device. In Subsection 4.2, we present a state-of-the-art algorithm for classification and pose estimation, demonstrating how it benefits from using our synthetic data for supervised 2.5D recognition.

4.1. Depth Error Evaluation

To validate the correctness of our simulation pipeline, we first replicate the set of experiments used by Landau *et al.* [20, 21], to compare the depth error induced by the proposed simulation tool to experimental values, as well as to the results from Landau *et al.* solution [20, 21], from *BlenSor* [13] and from three depth error models for Microsoft Kinect from the literature—respectively developed by Menna *et al.* [27], Nguyen *et al.* [30] and Choo *et al.* [6, 20]. For all the synthetic and experimental datasets, we use a flat surface placed in front of the depth sensor at various known distances, and at various tilt angles to the focal plane. For each position, several images are captured. The experimental dataset with ground truth was kindly provided by Landau [20, 21].

Figure 8(A) shows the influence of the distance between the plane and the sensor on the standard depth error of the resulting scans. The trend in our synthetic data matches well the one observed in experimental scans, along with the model from Choo *et al.* (recalibrated by Landau on the same experimental data [20]). As noted by Landau *et al.* when defending their own simulation tool, these models have been developed based on experimental results which are inherently correlated to the characteristics of their envi-

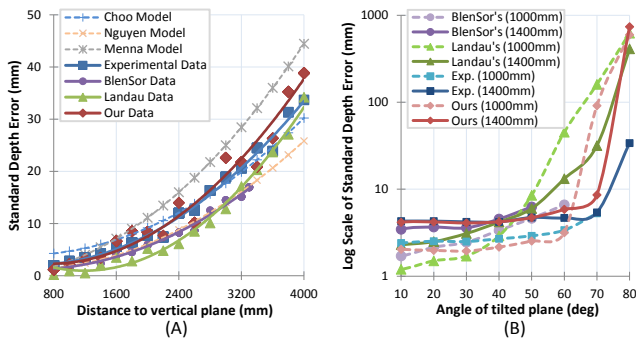


Figure 8. Standard depth error (in mm) as a function of (A) the distance (in mm) to a vertical flat wall, and (B) its tilt angle (in deg) for various fixed distances.

ronment and sensor. We could expect other data not to perfectly align with such models (as proved by the discrepancies among them). We can still notice that the quality of our synthetic images degenerates slightly more for larger distances than the experimental one, which may benefit from a better—patented—algorithm for the depth reconstruction [?]; though our method behaves overall more realistically than Landau’s one or *BlenSor*.

In Figure 8(B), we evaluate how our synthetic device fares when gradually tilting the plane from orthogonal to almost parallel to the optical axis, comparing to the experimental data provided by Landau and to other methods. Here again, the standard errors induced by our tool match closely the experimental results for tilt angles below 70° , with some overestimation for steeper angles but a similar trend, unlike the other methods. It should however be noted that for such incident angles, both real scans and those generated by our solution have most of the depth information missing, due to the poor reflection and stretching of the projected pattern(s), heavily impairing the depth reconstruction.

As a final experiment related to the error modeling, we compute the standard depth error as a function of the radial distance to the focal center. As previously observed, Figure 9 shows us that our pipeline behaves more similarly to the real device than the other tools, despite inducing slightly more noise for larger distances and thus more distorted pattern(s). As observed in all three figures, *DepthSynth* however satisfyingly reproduces the oscillating evolution of the noise when increasing the distance and reaching the edges of the scans—a well-documented phenomenon caused by “wiggling” and distortion of the pattern(s) [10, 18].

4.2. Application to the 6-DOF Pose Estimation and Classification Problems

Among the applications which can benefit from our pipeline, we formulate a 6-DOF camera pose recognition and classification problem from a single 2.5D image into an image retrieval problem. During the training stage, we discretize N_p camera poses, generate the synthetic 2.5D image for each pose and each object using the proposed pipeline, and encode each picture via a low dimension image representation with its corresponding camera pose and class. We build this way a database for pose and class retrieval problems. Given an unseen image, its representation is thus used to get its index in the saved database. K-nearest neighbor search implemented with KD-Tree is used to fulfill the indexing step.

In this framework, two components play an important role in ensuring a successful indexing. One is the 2.5D image simulation, and the other one is the image representation. A data simulation process is considered as fitting when it minimizes the quality gap between synthetic and acquired depth data—a minimization we achieved as explained in

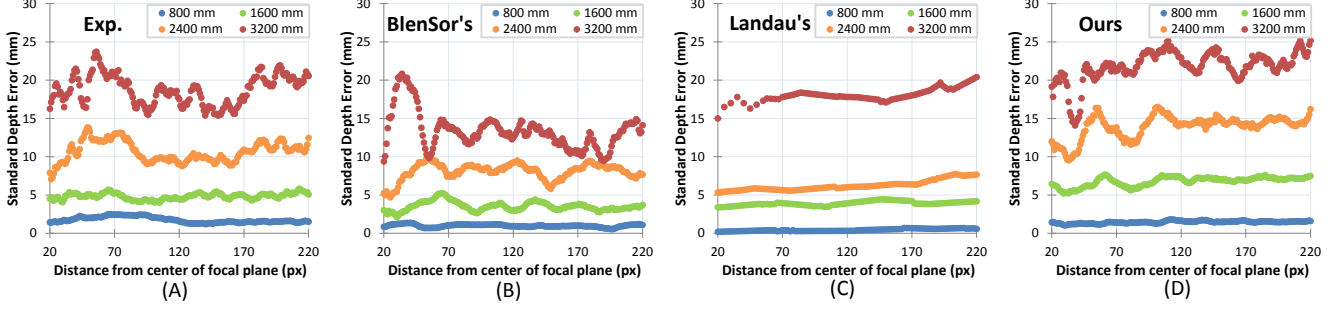


Figure 9. **Standard depth error (in mm) as a function of the radial distance (in px) to the focal center**, plotted for the (A) experimental data, and for the synthetic data from (B) *BlenSor*, (C) Landau’s pipeline and (D) *DepthSynth*, for various fixed distances.

Section 3. On the other hand, a fitting image representation should carry discriminative pose and class information, resulting in successful searches. The degree of discrimination for recognition problems can be defined as the distance between two image representations, which should be small when the objects are similar and their camera poses close to each other, and respectively big when the objects and/or poses are different.

To demonstrate the advantages of using the realistic data generated by our pipeline irrespective of the selected features, we present a method using case-specific *computer-crafted* representations generated by a CNN, before detailing our experiment and discussing its results.

4.2.1 CNN-based Image Representation

Recently, CNN-based approaches demonstrated a significant performance boost on image classification and feature learning tasks. In their work, Wohlhart *et al.* [43] proposed such a feature learning approach for classification and pose estimation. Their studies show promising results using synthetic data without structured noise simulation. Motivated by their work, we opted for a LeNet structure [22] with custom hyper-parameters, in order to learn discriminative 2.5D depth image representations. The proposed CNN structure is illustrated in Figure 10. The output layer of our network is used as image representation. In order to guide the computer in its learning, the discriminative Euclidean distance is enforced, using the loss function presented in [43] over all the CNN weights \mathbf{w} :

$$L = L_{triplet} + L_{pairwise} + \lambda \|\mathbf{w}\|_2^2, \quad (4)$$

where $L_{triplet}$ is the triplet loss function, $L_{pairwise}$ the pairwise loss function, and λ the regularization factor to avoid over-fitting. A triplet is defined as (p_i, p_i^+, p_i^-) , with p_i one class and pose sampling point, p_i^+ a sampling point defined as *close* to p_i (similar class and/or pose) and p_i^- another one defined as *far* from p_i^+ (different class and/or pose). A pair is defined as (p_i, p_i') , with p_i one class and pose sampling point and p_i' its perturbation in terms of pose and noise conditions, to enforce proximity between the descriptors of

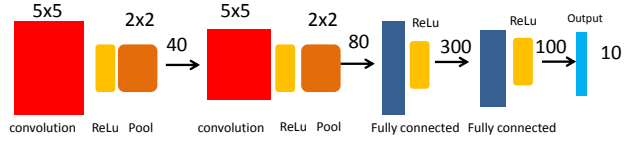


Figure 10. **LeNet CNN architecture [22] used in our approach.** It has two 5×5 convolution layers, each of them followed by a ReLu layer and a 2×2 Max pooling layer. Next, two fully connected layers lead to the output layer (also fully connected), used as image representation $f(\cdot)$.

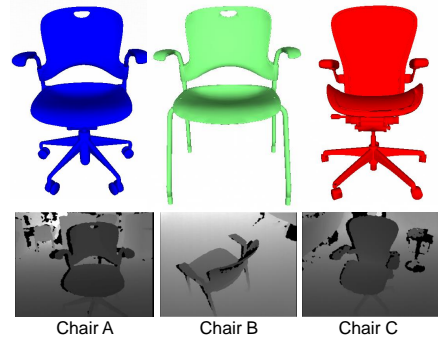


Figure 11. **CAD models and sample real images used in the experiments.** Note the strong similarities among these chairs.

similar images despite different imaging conditions. Given $f(\cdot)$ the CNN-generated representation and m a margin, $L_{pairwise}$ is defined as the sum of the squared Euclidean distances between $f(p_i)$ and $f(p_i')$, and $L_{triplet}$ as:

$$L_{triplet} = \sum_{(p_i, p_i^+, p_i^-)} \max(0, 1 - \frac{\|f(p_i) - f(p_i^-)\|_2}{\|f(p_i) - f(p_i^+)\|_2 + m}), \quad (5)$$

Using *Caffe* [15], our network has its weights optimized by stochastic gradient descent on mini-batches.

4.2.2 Data Preparation

As target models for the experiment, we select 3 similar-looking office chairs and download their CAD models from the manufacturers’ websites (Figure 11). The following procedure is performed to capture the real 2.5D dataset and

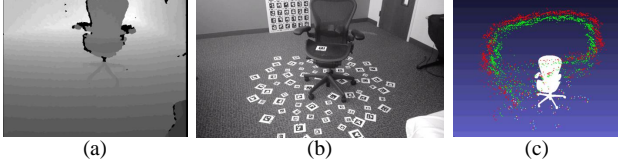


Figure 12. **Real data acquisition and processing.** (a) Sample real depth data. (b) Sample RGB data, with markers. (c) Recovered trajectory and poses for each camera location as well as markers for Chair C.

its ground-truth pose annotations: AR markers are placed on the floor around each chair, an Occipital Structure sensor is mounted on a tablet, and its IR camera is calibrated according to the RGB camera of the tablet. An operator holds the tablet, roughly pointing at the chair and walks around it while the depth sensor and RGB camera are continuously capturing sequences of RGBD frames.

In a comprehensive and redundant annotation procedure, we manually generate 2D-3D correspondences on chairs regions based on visual landmarks, choosing a representative set of approximately 60 frames. These landmarks are fed into a camera pose estimation procedure using robust Direct Linear transform (DLT). A next step considers the estimated camera poses and the detected 2D locations of markers to generate triangulated 3D markers locations in the CAD coordinate system. However, given their movable parts, the actual chairs deviate from their model in many ways. We thus iteratively reduce the deviation for the final ground-truth sequence by verifying reprojections of the model into the aligned RGB frames, and by additionally verifying the consistency of the triangulated markers positions relative to the chairs elements. The calibration parameters for the IR and RGB camera are then used to align the depth images into the common 3D coordinate system. In a final fine-tuning step, the annotated poses and CAD models are fed into the simulation pipeline, to generate the corresponding synthetic noise-free depth maps. By running an Iterative Closest Point (ICP) method [2] to align these synthetic scans to the real ones, we optimize the ground-truth 3D poses for our real testing dataset (which will be made publicly available), as shown in Figure 12.

4.2.3 Evaluation on Pose Estimation

As a first experiment, we limit the aforementioned approach to pose estimation only, training and testing it over the data of a single chair (Chair C). For the CNN training, 30k synthetic depth scans are used to form 100k samples (triplets + pairs). The learned representation is then applied for the indexation of all the 30k images, using FLANN [28]. For testing, the representations of the 1024 depth images forming the real dataset are extracted and indexed. For each, the pose of the best search result is then sent to the simulation

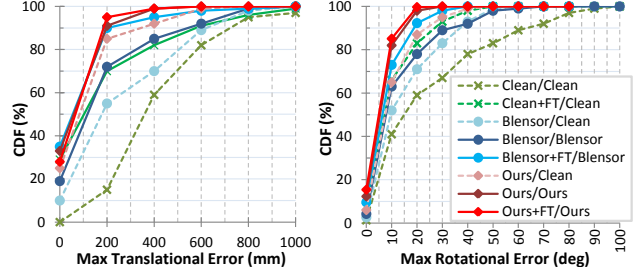


Figure 13. **Cumulative Distribution Function (CDF) on errors in rotation and translation for pose estimation on the Chair C dataset**, using the presented recognition method trained over different synthetic datasets.

pipeline, to render a synthetic depth image. Using ICP, it is aligned to the input scan to refine the 3D pose, considered as the final estimation.

To demonstrate how the quality of the synthetic training data impacts the estimation, three different synthetic datasets are generated, respectively using clean 3D rendering for noiseless depth data, *Blensor*, and *DepthSynth*. Each dataset is used either for both representation-learning and indexing; or only for learning, with the clean dataset used for indexing. We also further apply some fine-tuning (FT) to the CNN training, feeding it with 200 real scans from the testing dataset, forming 3k samples. Estimated 3D poses are compared to the ground-truth poses, and the Cumulative Distribution Functions (CDFs) on errors in rotation and translation are shown in Figure 13.

This figure reveals how the method trained over our synthetic datasets gives consistently better results on both translation and rotation estimations, furthermore not gaining much in accuracy after fine-tuning—a step supposedly “cheating” by adding real data from the testing dataset to the training.

4.2.4 Evaluation on Classification

We consider in a second time the classification problem for the 3 chairs. Using the same synthetic training datasets extended to all 3 objects, we evaluate the accuracy of the recognition method over a testing dataset of 1024 real depth images for each chair, taking as final estimation the class of the nearest neighbor when searching the database with the extracted representation for each image.

Given the difficulty of this task due to the strong similarities among the objects, the recognition method is performing quite well, as shown in Table 2. Again, it can be seen that it gives consistently better results when trained over our synthetic data; and that unlike other training datasets, ours doesn’t gain much from the addition of real data, which validates its inherent realism.

Table 2. **Classification results** over the *3-Chairs* dataset, using the recognition method trained over different synthetic datasets.

Data for CNN-Training	Data for Indexing	Error (%)
Clean	Clean	37.1
Clean + fine tuning	Clean	23.5
BlenSor	Clean	29.8
BlenSor	BlenSor	25.3
BlenSor + fine tuning	BlenSor	16.4
DepthSynth	Clean	18.5
DepthSynth	DepthSynth	13.6
DepthSynth + fine tuning	DepthSynth	12.3

5. Conclusion and Future Work

We presented *DepthSynth*, a pipeline to generate large depth image datasets from 3D models, simulating the mechanisms of a wide panel of structured light sensors to achieve unique realism with minimum effort. We demonstrated how our work can be used as input for the training of common 2.5D recognition tasks, outperforming the original results of state-of-the-art methods using lower-quality training data.

We thus believe this concept will prove itself greatly useful to the computer vision community, leveraging the parallel efforts to gather detailed 3D model datasets. The ability to generate realistic synthetic data and corresponding ground truth can promote a large number of data-driven algorithms covering depth-based applications, by providing the training or benchmarking resources they need.

Despite the maturity of our proposed concept, we are well aware—through the previous results and our own advanced use cases—it has room for improvement, such as more sophisticated background modeling, occlusion handling and enhanced reflectance acquisition (e.g. as a Bidirectional Reflectance Distribution Function (BRDF)).

References

- [1] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze. Point cloud library. *IEEE Robotics & Automation Magazine*, 1070(9932/12), 2012.
- [2] P. Besl and D. McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.
- [3] R. A. Brooks, R. Creiner, and T. O. Binford. The acronym model-based vision system. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI’79, pages 105–113. Morgan Kaufmann Publishers Inc., 1979.
- [4] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.
- [5] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung. On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pages 223–232. Wiley Online Library, 2003.
- [6] B. Choo, M. Landau, M. DeVore, and P. A. Beling. Statistical analysis-based error models for the microsoft kinecttm depth sensor. *Sensors*, 14(9):17430–17450, 2014.
- [7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [8] M. F. Fallon, H. Johannsson, and J. J. Leonard. Point cloud simulation & applications, 2012. <http://www.pointclouds.org/assets/icra2012/localization.pdf>. Accessed: 2015-09-23.
- [9] S. Fidler, S. Dickinson, and R. Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *NIPS*, pages 611–619, 2012.
- [10] P. Fursattel, S. Placht, M. Balda, C. Schaller, H. Hofmann, A. Maier, and C. Riess. A comparative error analysis of current time-of-flight sensors. *IEEE Transactions on Computational Imaging*, 2(1):27–41, 2016.
- [11] A. Geiger, M. Roser, and R. Urtasun. Efficient large-scale stereo matching. In *ACCV*, pages 25–38, 2011.
- [12] S. Gold, A. Rangarajan, C. ping Lu, and E. Mjolsness. New algorithms for 2d and 3d point matching: Pose estimation and correspondence. *Pattern Recognition*, 31:957–964, 1997.
- [13] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree. Blensor: blender sensor simulation toolbox. In *Advances in Visual Computing*, pages 199–208. Springer, 2011.
- [14] K. E. A. O. J. Xiao, J. Hays and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer Vision and Pattern Recognition, 2010. CVPR 2010. IEEE Conference on*, pages 3485–3492. Springer, 2010.
- [15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
- [16] K. Konolige. Small vision systems: Hardware and implementation. In *Robotics Research*, pages 203–212. Springer, 1998.
- [17] K. N. Kutulakos and E. Steger. A theory of refractive and specular 3d shape by light-path triangulation. In *IEEE ICCV*, pages 1448–1455, 2005.
- [18] E. Lachat, H. Macher, M. Mittet, T. Landes, and P. Grussenmeyer. First experiences with kinect v2 sensor for close range 3d modelling. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5):93, 2015.
- [19] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *IEEE ICRA*, pages 1817–1824. IEEE, 2011.
- [20] M. J. Landau. *Optimal 6D Object Pose Estimation with Commodity Depth Sensors*. PhD thesis, University of Virginia, 2016. <http://search.lib.virginia.edu/catalog/hq37vn57m>. Accessed: 2016-10-20.
- [21] M. J. Landau, B. Y. Choo, and P. A. Beling. Simulating kinect infrared and depth images. 2015.

- [22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition.
- [23] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing ikea objects: Fine pose estimation. In *IEEE ICCV*, pages 2992–2999. IEEE, 2013.
- [24] Y. Ma, K. Boos, J. Ferguson, D. Patterson, and K. Jonaitis. Collaborative geometry-aware augmented reality with depth sensors. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 251–254. ACM, 2014.
- [25] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London B: Biological Sciences*, 200(1140):269–294, 1978.
- [26] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE IROS*, September 2015.
- [27] F. Menna, F. Remondino, R. Battisti, and E. Nocerino. Geometric investigation of a gaming active device. In *SPIE Optical Metrology*, pages 80850G–80850G. International Society for Optics and Photonics, 2011.
- [28] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014.
- [29] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '11*, pages 127–136, 2011.
- [30] C. V. Nguyen, S. Izadi, and D. Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 524–530. IEEE, 2012.
- [31] K. Rematas, T. Ritschel, M. Fritz, and T. Tuytelaars. Image-based synthesis and re-synthesis of viewpoints guided by 3d models. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3898–3905. IEEE, 2014.
- [32] A. Rozantsev, V. Lepetit, and P. Fua. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, 2015.
- [33] A. Saxena, J. Driemeyer, and A. Y. Ng. Learning 3-d object orientation from images. In *IEEE ICRA*, pages 4266–4272, 2009.
- [34] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from a single depth image. In *IEEE CVPR*, June 2011.
- [35] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgb-d images. In *ECCV*, pages 746–760. Springer, 2012.
- [36] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *IEEE ICRA*, pages 509–516. IEEE, 2014.
- [37] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *NIPS*, pages 665–673, 2012.
- [38] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *IEEE CVPR*, pages 567–576, 2015.
- [39] M. Stark, M. Goesele, and B. Schiele. Back to the future: Learning shape models from 3d cad data. In *BMVC*, pages 106.1–106.11. BMVA Press, 2010.
- [40] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *IEEE ICCV*, 2015.
- [41] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for CNN: viewpoint estimation in images using cnns trained with rendered 3d model views. *CoRR*, abs/1505.05641, 2015.
- [42] Y. Wang, J. Feng, Z. Wu, J. Wang, and S.-F. Chang. From low-cost depth sensors to cad: Cross-domain 3d shape retrieval via regression tree fields. In *ECCV*, September 2014.
- [43] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *IEEE CVPR*, pages 3109–3118, 2015.
- [44] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE CVPR*, pages 1912–1920, 2015.
- [45] R. Zhang, P.-S. Tsai, J. Cryer, and M. Shah. Shape-from-shading: a survey. *IEEE TPAMI*, 21(8):690–706, 1999.